UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #8 – *C* Expressions (1)

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Administrative Issues

- Lab#3 starting Today (Monday, Feb. 6), and due **5pm, Wednesday, Feb. 8**

- Lab#2 grade is available from M: drive
  - Please send your grade concern to TA Peter (glv@umassd.edu) and cc to me if there is any

- Homework #2
  - Due **Friday, Feb. 10**
  - Please follow the "submission guidelines" available in the course website to submit your answers to your name folder at the class M: drive if you haven't
  - Late submission is subject to penalty.

Review of Lecture #7

- Formatted input function scanf()
  - Function format
  - Rules
  - Examples
- Common programming errors

Do not forget to put the & in front of each variable in the address list of scanf()!!!

Do not put & in front of a variable in the data list of printf()!!!

Topics

- Overview of expressions
- Five types of expressions in C

C Expressions

- Expressions are things that have values
  - A variable by itself is an expression: radius
  - A constant by itself is an expression: 3.14

- Often expressions are combinations of operands (data that take part into operation: variables, constants) and operators (+, -, *, etc):

```
3.14 * radius * radius

2+3
```

Expression Evaluation

- The value of an expression depends on the data types and values, and on the operators used.
- Expressions always reduce to a single value!

Why Study Expressions?

- We need precise rules that define exactly what an expression means:

  What is the value of 10 - 2 * 2+ 2?

- Arithmetic on a computer may differ from everyday arithmetic or math:

  2 / 3 is zero in C, not 0.666667 !

Types of operators

- Binary: operates on two operands

  *2*a*

  a+ b

  3/2

- Unary: operates on one operand

  -23.4

Types of Expressions

- Primary expressions
- Binary expressions
- Assignment expressions
- Postfix expressions
- Unary expressions

Type 1: Primary Expressions

- Consist of only one operand with no operator

- The operand can be a
  - name: any identifier for a variable or a function
    - E.g.: student_age, temperature, tax
  - constant: data whose value cannot change during the execution of the program
    - E.g.: 7, 23.6, 'a', "hello"
  - parenthetical expression: any (complex) expression enclosed in parentheses
    - E.g.: (2*3+7)

Type 2: Binary Expressions

- Formed by an *operand-operator-operand* combination
  - Multiplicative expressions
  - Additive expressions

Multiplicative Expressions

- The multiply,divide and modulus operators have the highest **priority** among binary expressions.
- They are therefore evaluated first among binary expressions.

- Example: multiply (*)
  - z = 1 + 2*3 = ??
  - z = 1 + 2*3 = 1 + (2*3) =7

Division

- If both operands are integers, then the result of the division is the integral value of the quotient, expressed as an integer.

- If either operand is a floating-point number, then the result of the division is a floating-point number with a type that matches the higher format of the operands (float, double, long double).

Note!

- To get a float as a result of a division, at least one of the operands must be float and the result must be assigned to a float.

Exercise

What is the result of:

- 2/3
- 2.0/3
- 2.0/3.0

Modulo Operation

- % symbolizes the modulo operation
- Modulo divides the first operand by the second operand and returns the remainder.
- Both operands must be integer types.
- Operator returns the remainder as an integer type.

Exercise

- What is the result of 2/5 and 2%5?

- What is the result of 5/2 and 5%2?

Summary of Multiplicative Expressions

*	Result is algebraic multiplication of two operands
/	Result is algebraic division of first operand by second operand • Integer quotient if both operands are integer • Floating-point quotient if either operand is a floating-point number
%	Result is integer remainder after first operand is divided by second operand. Both operands must be integer types

Additive Expressions

- Additive operators: + (add) - (subtract)
- Example: tax + 17

  temperature – 3

- Additive expressions are evaluated after multiplicative expressions
- Example:
  - z = 2 + 7/3;
  - z = 2 + 7/3 = 2+ (7/3) = ?;

Types of Expressions (Agenda)

- ✓ Primary expressions
- ✓ Binary expressions:
  - Multiplicative expressions
  - Additive expressions
- Assignment expressions
- Postfix expressions
- Unary expressions

Type 3: Assignment Expressions

- Assignment expressions evaluates the operand on the right side of the operator (=) and place its value in the variable on the left

- Two types
  - Simple assignment
  - Compound assignment

Simple Assignment

- The left operand in an assignment expression must be **a single variable**!

```
b = 1;
k = a+b;
```

Compound Assignment

- A shorthand notation for a simple assignment.
- It requires the left operand be repeated as a part of the right expression
- Examples:

  x *= y; is equivalent to x = x*y;

  x /= y; is equivalent to x = x /y;

  x %= y; is equivalent to x=x%y;

  x += y; is equivalent to x = x+y;

  x -= y; is equivalent to x = x-y;

Exercise

- Assume int x=3; What is the result of the following expressions?

```
x *=  2;
x /=  4;
x %= 4;
x += 9;
```

Note!

- If a compound statement is used with a binary expression, the binary expression is evaluated first.

- If x = 2 and y =3 what is the value of

```
x *=  y + 7 ;
```

x = x*(y+7) = 20

Types of Expressions (Agenda)

- ✓ Primary expressions
- ✓ Binary expressions:
  - Multiplicative expressions
  - Additive expressions
- ✓ Assignment expressions
- Postfix expressions
- Unary expressions

Type 4: Postfix Expressions

- It consists of one operand followed by one operator

- Example:
  - Postfix increment: b++;

    /* b++ has the same effect as b = b+1;*/
  - Postfix decrement: b--;

    /* b-- has the same effect as b=b-1; */

Note!

- The value of the postfix increment expression is determined before the value of the variable is increased.
- Example:

```
int a=3;
x=a++;
x=3     a=4

Therefore, x=a++; is equivalent to :

x=a;
a=a+1;
```

Note!

- The value of the postfix decrement expression is determined before the value of the variable is decreased.
- Example:

```
int b=7;
x = b--;
x=7     b=6
```

Therefore, x=b--; is equivalent to :

```
x=b;
b=b-1;
```

Exercise

What is the output of the printf()?

```
int b;
b =20;
printf("value of b: %2d\n", b);
printf("value of b++: %2d\n", b++);
printf("value of b: %2d\n", b);
```

Types of Expressions (Agenda)

- ✓ Primary expressions
- ✓ Binary expressions:
  - – Multiplicative expressions
  - – Additive expressions
- ✓ Assignment expressions
- ✓ Postfix expressions
- • Unary expressions

Type 5: Unary Expressions

- It consists of one operator and one operand
- Example operators
  - Prefix increment/decrement
  - sizeof
  - plus/minus

Prefix Operators

- Prefix increment: ++b;

  /* ++b has the same effect as b = b+1;*/

- Prefix decrement: --b;

  /* --b has the same effect as b=b-1; */

Note!

- The effect of a prefix operator takes place before the expression that contains the operator is evaluated.

```
x=++a;
```

is equivalent to

```
a=a+1;
x=a;
```

```
x=--b;
```

is equivalent to

```
b=b-1;
x=b;
```

In general, if the ++ or -- is before the operand, the addition or subtraction takes place before the value of the operand is determined.

If it is after the operand, the addition or subtraction takes place after the value of the operand is determined.

Exercise

What is the output of printf()?

```
int b;
b = 20;
printf(“ value of b: %2d\n”, b);
printf(“value of ++b: %2d\n”, ++b);
printf(“value of b : %2d\n”, b);
```

The *sizeof* Operator

- It tells you the size in bytes of the operand.
- Example:

a = sizeof(int); → machine dep., can be 2 or 4 or larger

b = sizeof(float); → 4

e = sizeof(0.32); → 8 (the default type is double, which has 8 bytes.)

Unary Plus/Minus

- They are simply plus and minus sign
- In C, they are operators
  - The plus + does nothing but yield the value of the operand
  - The minus – is used to change the sign of a value algebraically: from plus to minus or minus to plus

- Note: the value of the stored variable (operand) is unchanged!

Exercise

What is the output of the printf()?

```
int b;
b =7;
printf("value of +b: %d\n", +b);
printf("value of -b: %2d\n", -b);
printf("value of b: %2d\n", b);
```

Review Questions

- What is the output of each printf() statement in the program?

```
#include <stdio.h>
void main(void)
{
  int a=3;
  int b=7;
  float c=6.0;
  printf("%d\n", a/b);
  printf("%f\n", a/c);
  printf("%d\n", b/a);
  printf("%f\n", c/a);
  printf("%d\n", a%b+a);
  printf("%f\n", a%c);
  printf("%d\n", b%a);
  b=a++;
  printf("%d\n", b);
  printf("%d\n", a);
  printf("%d\n", a--);
}
```

Summary of Lecture #8

- Expressions are combinations of operands (data that take part into operation: variables, constants) and operators (+, -, *, etc)
- Five types of expressions in C
  - Primary expressions
  - Binary expressions:
    - Multiplicative expressions
    - Additive expressions
  - Assignment expressions
  - Postfix expressions
  - Unary expressions

Things To Do

- Homework #2
  - Due by Friday, Feb. 10

Next Topic

- Expressions (Cont'd)